Chapter 6 Worksheet

Vector class:

```
import greenfoot.Greenfoot;

public final class Vector
{
    double dx;
    double dy;
    int direction;
    double length;

    /** Create a new, neutral vector. */
    public Vector(){ }

    /**
     * Create a vector with given direction and length. The direction should be in
     * the range [0..359], where 0 is EAST, and degrees increase clockwise.
     */
    public Vector(int direction, double length){
        this.length = length;
        this.direction = direction;
        updateCartesian();
    }

    /** Create a vector by specifying the x and y offsets from start to end points. */
    public Vector(double dx, double dy){
        this.dx = dx;
        this.dy = dy;
        updatePolar();
    }

    /** Set the direction of this vector, leaving the length intact. */
    public void setDirection(int direction){
        this.direction = direction;
        updateCartesian();
    }

    /** Add another vector to this vector. */
    public void add(Vector other) {
        dx += other.dx;
        dy += other.dy;
        updatePolar();
    }

    /** Set the length of this vector, leaving the direction intact. */
    public void setLength(double length) {
        this.length = length;
        updateCartesian();
    }

    /**
     * Scale this vector up (factor > 1) or down (factor < 1). The direction
     * remains unchanged.
     */
    public void scale(double factor) {
        length = length * factor;
        updateCartesian();
    }

    /** Set this vector to the neutral vector (length 0). */
    public void setNeutral() {
        dx = 0.0;
        dy = 0.0;
        length = 0.0;
        direction = 0;
```

```
    }

    /** Revert to horizontal component of this movement vector. */
    public void revertHorizontal() {
        dx = -dx;
        updatePolar();
    }

    /** Revert to vertical component of this movement vector. */
    public void revertVertical() {
        dy = -dy;
        updatePolar();
    }

    /** Return the x offset of this vector (start to end point). */
    public double getX() { return dx; }

    /** Return the y offset of this vector (start to end point). */
    public double getY() { return  dy; }

    /** Return the direction of this vector (in degrees). 0 is EAST. */
    public int getDirection() { return direction; }

    /** Return the length of this vector. */
    public double getLength() { return length; }

    /** Update the direction and length from the current dx, dy. */
    private void updatePolar()
    {
        this.direction = (int) Math.toDegrees(Math.atan2(dy, dx));
        this.length = Math.sqrt(dx*dx+dy*dy);
    }

    /** Update dx and dy from the current direction and length. */
    private void updateCartesian()
    {
        dx = length * Math.cos(Math.toRadians(direction));
        dy = length * Math.sin(Math.toRadians(direction));
    }
}
```

SmoothMover class:

```
import greenfoot.*;  // (World, Actor, GreenfootImage, and Greenfoot)

public abstract class SmoothMover extends Actor
{
    private Vector movement;
    private double exactX;
    private double exactY;

    public SmoothMover() { this(new Vector()); }

    /** Create new thing initialised with given speed. */
    public SmoothMover(Vector movement) { this.movement = movement; }

    /** Move in the current movement direction. */
    public void move() {
        exactX = exactX + movement.getX();
        exactY = exactY + movement.getY();
        super.setLocation((int) exactX, (int) exactY);
    }
```

Chapter 6 Worksheet

```
    /** Set the location using exact (double) co-ordinates. */
    public void setLocation(double x, double y) {
        exactX = x;
        exactY = y;
        super.setLocation((int) x, (int) y);
    }

    /**
     * Set location. Redefinition of the standard Greenfoot method to make sure
     * the exact co-ordinates are updated in sync.
     */
    public void setLocation(int x, int y) {
        exactX = x;
        exactY = y;
        super.setLocation(x, y);
    }

    /** Return the exact x co-ordinate (as a double). */
    public double getExactX() { return exactX; }

    /** Return the exact y co-ordinate (as a double). */
    public double getExactY() { return exactY; }

    /** Modify the current movement by adding a new vector to the existing movement. */
    public void addForce(Vector force) { movement.add(force); }

    /**
     * Accelerate the speed of this mover by the given factor. (Factors less than 1 will
     * decelerate.) The direction remains unchanged.
     */
    public void accelerate(double factor) {
        movement.scale(factor);
        if (movement.getLength() < 0.15) {
            movement.setNeutral();
        }
    }

    /** Return the speed of this actor. */
    public double getSpeed() { return movement.getLength(); }

    /** Return the current movement of this object (as a vector). */
    public Vector getMovement() { return movement; }
}
```

(1) Make the changes in Rocket so that the Rocket moves whenever act() is called:

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class Rocket  extends SmoothMover
{
    public void act()
    {

    }
}
```

(2) What changes are needed to the move method of SmoothMover to check for edges as the actor is moving within the world and wrap around to the other side.

```
public void move() {
     exactX = exactX + movement.getX();
     exactY = exactY + movement.getY();









     super.setLocation((int) exactX, (int) exactY);
}
```

(3) Write the code in checkForKeyPress() to have the Rocket turn when the user presses the left and right arrow keys.

```
private void checkForKeyPress() {









}
```

Chapter 6 Worksheet

(4) Write the code for the checkForBarrels method in the Rocket

```
private void checkForBarrels() {









}
```

(5) Write the line of code that retrieves all of the Barrels from the world and saves them into a variable named barrels.

(6) Write the for-each loop that iterates over each Barrel in the collection named barrels and moves each barrel down 5 pixels.

ANSWER to Question 6:

```
for(Barrel b: barrels) {
      b.setLocation(b.getX(), b.getY() + 5);
}
```

Chapter 6 Worksheet

ANSWER to Question 1:

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class Rocket  extends SmoothMover
{
    public Rocket() {
        super(new Vector(0, 1.0));
    }

    public void act()
    {
        move();
    }
}
```

ANSWER to Question 2:

```
public void move() {
        exactX = exactX + movement.getX();
        exactY = exactY + movement.getY();

        if(exactX <= 0) {
            exactX = getWorld().getWidth() - 10;
        }
        else if(exactX >= getWorld().getWidth()) {
            exactX = 10;
        }

        if(exactY <= 0) {
            exactY = getWorld().getHeight() - 10;
        }
        else if(exactY >= getWorld().getHeight()) {
            exactY = 10;
        }

        super.setLocation((int) exactX, (int) exactY);
}
```

ANSWER to Question 3:

```
private void checkForKeyPress() {
        if(Greenfoot.isKeyDown("left")) {
            setRotation(getRotation() - 1);
            getMovement().setDirection(getRotation());
        }
        else if(Greenfoot.isKeyDown("right")) {
            setRotation(getRotation() + 1);
            getMovement().setDirection(getRotation());
        }
}
```

ANSWER to Question 4:

```
private void checkForBarrels() {

      if(getOneIntersectingObject(Barrel.class) != null) {
            setRotation(getRotation() - 180);
            getMovement().setDirection(getRotation());
      }
}
```

ANSWER to Question 5:

```
java.util.List<Barrel> barrels = getWorld().getObjects(Barrel.class);
```